View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Kyoto University Research Information Repository

KURENAI
Kyoto University Research Information Repository

KYOTO UNIVERSITY

Title	非同期通信に基づく並列処理言語の表示的意味記述について(計算アルゴリズムと計算量の基礎理論)
Author(s)	那須, 隆; 大山口, 通夫
Citation	数理解析研究所講究録 (1989), 695: 263-272
Issue Date	1989-06
URL	http://hdl.handle.net/2433/101382
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

Kyoto University

数理解析研究所講究録
695巻 1989年 263-272

非同期通信に基づく並列処理言語の表示的意味記述について

三重大学工学部　那須　隆　　(Takashi Nasu　)

三重大学工学部　大山口　通夫　(Michio Oyamaguchi)

1.はじめに

非決定的選択の機能を持つ並列処理言語の意味記述においては通信の時間的前後関係をなんらかの方法で表すことの必要性が、Brock,Ackerman[2]によって指摘された。しかし、時間をあまり陽に表して実現系の詳細に触れることは好ましくない。そこで、筆者らは実用的な並列処理言語に対する簡潔で抽象度の高い意味記述を与えることを目的として、比較的単純な言語を設計し、この言語に対する表示的な意味記述を考察した。本稿では、この言語と通信の時間的前後関係の記述を必要最小限に制限した表示的意味記述の概要について報告する。

2.言語仕様

意味記述を考察した言語は、非同期通信，プロセス並列呼び出し，および非決定的選択の機能を持っている。この言語は、Broyらが[3]で与えた言語に動的なプロセス生成やローカル変数の宣言等の新しい機能を加え、実用的な言語に近いものにした。枠組みとしては、[1]に非決定的選択を加えたものに近い。図1にこの言語の構文をBNFを用いて示す。

```
<ﾌﾟﾛｸﾞﾗﾑ> ::= program <id><parm>;<ﾌﾟﾛｾｽ宣言>; … ;<ﾌﾞﾛｯｸ>.
<ﾌﾟﾛｾｽ宣言> ::= process <id><parm>;<ﾌﾞﾛｯｸ> endprocess
<ﾌﾞﾛｯｸ> ::= var <parmlist>;begin <文>; … end
<parm> ::=  ()|( <parmlist> )
<parmlist> ::= <id>,… :<型>|<id>, … :<型>;<pramlist>
<文> ::= ε |begin <文>;… end|<var-id> := <式>
      |if <式> then <文>|if <式> then <文1> else <文2>
      |while <式> do <文>|repeat <文> until <式>
      |<inpch-id>?<var-id>                    (* チャネルからの入力 *)
      |<outch-id>!<式>                        (*  チャネルへの出力  *)
      |chan <inpch-id> <- <outch-id>:<文>     (*    チャネル結合    *)
      |<process-id><parm>                     (*  プロセス呼び出し  *)
      |<文1> // <文2> // …                    (*      並列実行      *)
      |<文1> [] <文2> [] …                    (*    非決定的選択    *)
```

図１．構文の概略

プログラムは、プロセス宣言の列と初期プロセスを表すブロックからなる。プロセス宣言は仮引き数の宣言とブロックからなり、ブロックはローカル変数宣言と文の連なりからなる。文には、通常のプログラミング言語と同様な複文，代入文，条件文(if文)，繰り返し文(while文,repeat文)の他にチャネル(通信路)を用いた通信に関するもの，プロセス呼び出し(生成)文，並列実行文，ならびに非決定的選択文がある。

通信には入力用と出力用のチャネル変数を用意し、その結合を指定することで通信路が確保される。この結合されたチャネル変数をプロセスに引き数として渡すことで、プロセス間の通信が可能になる。この通信は非同期で、通信路は無限バッファを仮定している。

プロセス呼び出し文によって、プロセスの実体は動的に生成され、その実行がなされる。プロセスが他のプロセスを生成するとき、生成したプロセスの終了まで待たされる。しかし並列実行文と組み合わせることで、同時に複数のプロセスを生成，実行することが可能である。

3.形式的意味

Brock-Ackermanの問題点とは、入出力の時間的関係を陽に記述しないならば、同じ入出力関係を持つプログラムでも、出力を入力にフィードバックをかけると異なる動作をする場合があり、プログラムの意味を単なる入出力関係で定義するのは不十分であることを示したことである。

本稿で提案する意味記述は、チャネル通信での入力待ち状態になったときには、その場所を記憶しておき，次に他のプロセスなどからのチャネルへの出力を得たとき、その待ち状態の場所以降を実行するように定義している。これを表示的意味論の手法で記述するために、入力待ちの発生の有無や場所の記憶を意味領域の中で定義するとともに、これらを扱ういくつかの補助関数(述語)を導入した。この入力待ちとなった場所を記憶し、実行を制御するような意味記述を与えることによって通信の時間的前後関係をあまり抽象性を失うことなく記述できた。この意味記述に用いた領域(意味領域)を図２，意味関数を図３に与える。

ローカル環境　$\rho \in$ Lenv = [id→Loc]

状態　$\sigma \in$ State = [Location→Value]

グローバル環境　$\vartheta \in$ Genv = [id→Proc_func]

Proc_func = $\bigcup_{m \geqq 0}$ [Location^{m}→(Lenv×State)→$\mathcal{P}$(Lenv×State)]　*1

ﾌﾟﾛｸﾞﾗﾑの意味関数　$\mathcal{P}rg \in$ [プログラム→Genv→(Lenv×State)→$\mathcal{P}$(Lenv×State)]

プロセス宣言の意味関数　$\mathcal{D}p \in$ [プロセス宣言→Genv→Genv]

ブロックの意味関数　$\mathcal{B}l \in$ [ブロック→Genv→(Lenv×State)→$\mathcal{P}$(Lenv×State)]

変数宣言の意味関数　$\mathcal{B} \in$ [変数宣言→(Lenv×State)→(Lenv×State)]

文の意味関数　$\mathcal{C}, \mathcal{C}c \in$ [文→Genv→(Lenv×State)→$\mathcal{P}$(Lenv×State)]

式の評価関数　$\mathcal{E} \in$ [式→Lenv→State→Value]

(*1 $\mathcal{P}$(Lenv×State)はローカル環境と状態の対のべき集合[4])

図２．意味領域

3.1.意味領域

ローカル環境(Lenv)は変数名からその記憶場所(Location)を与え、状態(State)はロケーションからその内容である値を与える連続関数とする。グローバル環境はプロセス名からそのプロセスの関数（Proc_func）を与える連続関数とする。プロセスの意味関数 Proc_func は、引き数のロケーションをあたえたときに、ローカル環境と状態の対からそのべき集合$\mathcal{P}$(Lenv×State)への連続関数とする。これは非決定的選択のため得られる複数の結果を集合で表すためである。このべき集合は、[4]のPower-domainと同様な集合とする。

文の意味関数は、文とグローバル環境が与えられたときに、ローカル環境と状態の対からそのべき集合への関数とする。ブロックやプログラムの意味関数も同様である。プロセス宣言の意味関数は、プログラムに書かれたプロセス宣言の列に従っ

て、プロセスの名前とその意味関数の対応をグローバル環境に埋め込む。

3.2.意味関数

図３において、$\sigma.l_0$（すなわち $\sigma(l_0)$）は、状態σのときに使用されていない記憶場所の集合を表すとしており、これを用いてローカル変数の記憶領域を一元的に管理している。

チャネル通信においてチャネルが空っぽで入力待ちとなったときには、その場所を記憶しておく。このため、チャネル入力文やプロセス呼び出し文には、あらかじめ一意のラベルが割り当てられていると仮定する。これらの文において入力待ちとなったとき、このラベルを入力チャネル変数，もしくはプロセス変数のフィールドwaitloc に格納して、その場所を記憶させる。また、σ.waitは状態σのときにそれまでの実行において入力待ちが発生したことを表すと定義している。

また次のような補助関数を定義している。

a) LABEL(S) は、文S中に存在するすべてのラベルからなる集合を与える。

b) SKIPρ(σ,LABEL(S)) は、文S中において待ち状態がないことを示す述語である。

c) Var(S) は、文Sで用いられている変数すべてを表わす。

d) ロケーションを扱う関数として、ロケーションを取り出す関数getloc，使用されたロケーションを取り除く関数removeloc，ロケーションの集合をn分割する関数partition_nを用いている。

4．おわりに

本稿で提案した並列処理言語に対する表示的意味記述は、高い抽象的レベルにおいて意味を与えることができた。また本稿で扱った言語は、並列処理言語の基本的機能である並列動作，プロセス間通信，非決定的選択ならびにプロセスとローカル変数の宣言を含んでいるので、データの型や制御構造を追加することによって実用的な言語に容易に拡張することができる。

参考文献

[1] G.Kahn D.B.MacQueen: "Coroutines and network of parallel processes", IFIP 77 (1977) pp.993-998

[2] J.D.Brock W.B.Ackerman: "Scenarios: A Model of Nondeterminate Computation", LNCS 107 (1981) pp.252-259

[3] M.Broy C.Lengauer: "On Denotational versus Predicative Semantics", technical report of UNIVERSITÄT PASSAU Aug.1987

[4] G.D.Plotkin: "A Power Domain Construction", SIAM J.Compt. Sept.(1976) pp.452-487

プログラムの意味関数 $\mathcal{P}rg$　　($\Delta_1;\cdots;\Delta_m$ はプロセス宣言の列, Θ はブロックとする)

$$\mathcal{P}rg[\![\texttt{program}\ \Delta_1;\cdots;\Delta_m;\Theta\ .\ ]\!]_g(\rho,\sigma) = \mathcal{B}l[\![\Theta]\!](\mathcal{D}p[\![\Delta_1;\cdots;\Delta_m]\!]g)(\rho,\sigma)$$

プロセス宣言の意味関数 $\mathcal{D}p$　　(Δ_i = **process** $\mathrm{P}_i(a_i);\Theta_i$ **endprocess**　$(1 \le i \le m)$　とする)

$$\mathcal{D}p[\![\Delta_1;\cdots;\Delta_m]\!]_g = g[\gamma_1/\mathrm{P}_1,\ldots,\gamma_m/\mathrm{P}_m]$$

$$(\gamma_1,\ldots,\gamma_m) = fix\big(\lambda(\gamma_1,\ldots,\gamma_m):(\lambda(\rho_1,\sigma_1):\lambda\mu_1:\mathcal{B}l[\![\Theta_1]\!]_{g'}(\rho_1[\mu_1/a_1],\sigma_1)\ ),$$

$$\vdots$$

$$(\lambda(\rho_m,\sigma_m):\lambda\mu_m:\mathcal{B}l[\![\Theta_m]\!]_{g'}(\rho_m[\mu_m/a_m],\sigma_m)\ )\ )$$

ただし $g' = g[\gamma_1/\mathrm{P}_1,\ldots,\gamma_m/\mathrm{P}_m]$

ブロックの意味関数 $\mathcal{B}l$　　(δ は **parmlist** , Γ は 文の列とする)

$$\mathcal{B}l[\![\texttt{var}\ \delta;\Gamma]\!]_g = \big(\lambda(\rho,\sigma):\mathcal{C}c[\![\Gamma]\!]_g(\rho',\sigma')\big)$$

ただし $(\rho',\sigma') = (\textbf{IF}\ \sigma.wait = false\ \textbf{THEN}\ \mathcal{B}[\![\text{var}\ \delta]\!](\rho,\sigma)\ \textbf{ELSE}\ (\rho,\sigma)\ )$

変数宣言の意味関数 $\mathcal{B}$　　(δ_1,δ_2 は **parmlist** , τ は型とする)

$$\mathcal{B}[\![\texttt{var}\ \delta_1;\delta_2]\!] = \big(\lambda(\rho,\sigma):\mathcal{B}[\![\text{var}\ \delta_2]\!](\mathcal{B}[\![\text{var}\ \delta_1]\!](\rho,\sigma))\ \big)$$

$$\mathcal{B}[\![\texttt{var}\ \mathrm{x}_1,\ldots,\mathrm{x}_n:\tau]\!] = \big(\lambda(\rho,\sigma):\mathcal{B}[\![\texttt{var}\ \mathrm{x}_2,\ldots,\mathrm{x}_n:\tau]\!](\mathcal{B}[\![\texttt{var}\ \mathrm{x}_1:\tau]\!](\rho,\sigma))\ \big)$$

$$\mathcal{B}[\![\texttt{var}\ \mathrm{x}:\tau]\!](\rho,\sigma) = (\lambda l_x:(\rho[l_x/\mathrm{x}],\sigma[removeloc(\sigma.l_0,l_x)/l_0,\perp/l_x]))(getloc(\sigma.l_0))$$

（この τ はプロセス名, **inpch, outch** 以外の型 ）

$$\mathcal{B}[\![\texttt{var c}:\texttt{inpch}]\!](\rho,\sigma) = (\lambda\ (l_d,l_w,l_l,l_c):(\ \rho[l_d/\mathrm{c},l_w/\mathrm{c.wait},l_l/\mathrm{c.waitloc},l_c/\mathrm{c.connect}],$$

$$\sigma[removeloc(\ \sigma.l_0,(l_d+l_w+l_l+l_c))/l_0,$$

$$\epsilon/l_d,\ false/l_w,\ \epsilon/l_l,\ false/l_c])\ )$$

$$(getloc^4(\sigma.l_0))$$

$$\mathcal{B}[\![\ \texttt{var d}:\texttt{outch}]\!](\rho,\sigma) = (\lambda l_d:(\rho[l_d/\mathrm{d}],\sigma[removeloc(\sigma.l_0,l_d)/l_0,\epsilon/l_d])\ )(getloc(\sigma.l_0))$$

$$\mathcal{B}[\![\texttt{var p}:\tau_p]\!](\rho,\sigma) = (\lambda\ (l_d,l_w,l_l,l_e):(\ \rho[l_d/\mathrm{p},l_w/\mathrm{p.wait},l_l/\mathrm{p.waitloc},l_e/\mathrm{p.env}],$$

$$\sigma[removeloc(\sigma.l_0,\ (l_d+l_w+l_l+l_e))/l_0,$$

$$\tau_p/l_d,\ false/l_w,\ \epsilon/l_l,\ \perp/l_e])\ )$$

$$(getloc^4(\sigma.l_0))\qquad\qquad(\tau_p\ \text{はプロセス名})$$

図３．意味関数（その１）

文の意味関数　$\mathcal{C}c$, $\mathcal{C}$　　(S は文, E は式, L_i はラベルを表わすとする)

$$\mathcal{C}c[\![\mathrm{S}]\!]_g = (\lambda(\rho,\sigma):\ \textbf{IF}\ \sigma.wait = false \vee \neg SKIP_\rho(\sigma, LABEL(\mathrm{S}))$$
$$\textbf{THEN}\ \mathcal{C}[\![\mathrm{S}]\!]_g(\rho,\sigma)\quad \textbf{ELSE}\ \{\ (\rho,\sigma)\ \}\ )$$

$$SKIP_\rho(\sigma, Lset) = \neg(\exists \mathtt{c} \in \mathrm{ICI} \cup \mathrm{PSI} : \sigma.\rho.\mathtt{c}.\mathtt{wait} = true\ \wedge\ \sigma.\rho.\mathtt{c}.\mathtt{waitloc} \in Lset\ )$$

(ただし ICI, PSI は入力チャネル変数, プロセス変数の名前の集合,

$SKIP_\rho(\sigma, LABEL(\mathrm{S}))$ が真 = 文 S 中には待ち状態が無い)

空文

$$\mathcal{C}[\![\ \mathtt{begin\ end}\ ]\!]_g = (\lambda(\rho,\sigma) : \{\ (\rho,\sigma)\ \})$$

複文

$$\mathcal{C}[\![\ \mathtt{begin}\ \mathrm{S}_1;\mathrm{S}_2;\ldots\ \mathrm{S}_n\ \mathtt{end}\ ]\!]_g = \mathcal{C}c[\![\mathrm{S}_1]\!]_g \circ \mathcal{C}[\![\ \mathtt{begin}\ \mathrm{S}_2;\ldots\ \mathrm{S}_n\ \mathtt{end}\ ]\!]_g \qquad (\circ \text{ は関数合成})$$

$$p \circ q = (\lambda(\rho,\sigma) : \{(\rho'',\sigma'')\ \mid \exists(\rho',\sigma') \in p(\rho,\sigma)\ , (\rho'',\sigma'') \in q(\rho',\sigma')\ \})$$

$$p,\ q,\ p \circ q \in [(\mathtt{Lenv} \times \mathtt{State}) \to \mathcal{P}(\mathtt{Lenv} \times \mathtt{State})]$$

条件文

$$\mathcal{C}[\![\mathtt{if}\ E\ \mathtt{then}\ \mathrm{S}_1\ \mathtt{else}\ \mathrm{S}_2]\!]_g = (\lambda(\rho,\sigma) : \textbf{IF}\ (\ \sigma.wait = false \wedge \mathcal{E}[\![E]\!]_\rho.\sigma = true)$$
$$\vee\ \neg SKIP_\rho(\sigma, LABEL(\mathrm{S}_1))\quad \textbf{THEN}\ \ \mathcal{C}[\![\mathrm{S}_1]\!]_g(\rho,\sigma)$$
$$\textbf{ELSE}\quad \mathcal{C}[\![\mathrm{S}_2]\!]_g(\rho,\sigma)\ )$$

代入文

$$\mathcal{C}[\![\mathtt{x} := E]\!]_g = (\lambda(\rho,\sigma) : \{\ (\rho, \sigma[\mathcal{E}[\![E]\!]_\rho.\sigma / \rho.\mathtt{x}])\ \})$$

チャネル入力文

$$\mathcal{C}[\![L_i :\ \mathtt{c?\ x}]\!]_g = (\lambda(\rho,\sigma) : \textbf{IF}\ \sigma.\rho.\mathtt{c} = \epsilon\ \ \textbf{THEN}\ \ \{(\rho,\ \sigma[true/wait,\ true/\rho.\mathtt{c}.\mathtt{wait},$$
$$L_i/\rho.\mathtt{c}.\mathtt{waitloc}])\}$$
$$\textbf{ELSE}\quad \{(\rho, \sigma[\ false/wait,\ false/\rho.\mathtt{c}.\mathtt{wait},$$
$$first(\sigma.\rho.\mathtt{c})/\rho.\mathtt{x},\ rest(\sigma.\rho.\mathtt{c})/\rho.\mathtt{c}])\}\ )$$

チャネル出力文

$$\mathcal{C}[\![\mathtt{d!}\ E]\!]_g = (\lambda(\rho,\sigma) : \{(\rho, \sigma[append(\sigma.\rho.\mathtt{d},\ \mathcal{E}[\![E]\!]_\rho.\sigma)/\rho.\mathtt{d}]\ \})$$

図３．意味関数（その２）

チャネル結合文

$$\mathcal{C}[\![\texttt{chan c} \leftarrow \texttt{d}: \mathrm{S}]\!]_g = fix(\lambda F : (\lambda(\rho_1,\sigma_1) : \mathcal{C}c[\![\mathrm{S}]\!]_g(\rho_1, \sigma_1[true/\rho_1.\texttt{c.connect}]))\circ$$
$$(\lambda(\rho_2,\sigma_2) : \textbf{IF } (\sigma_2'.wait = false \vee Isconnect(\rho_2,\sigma_2'))$$
$$\textbf{THEN } \{ (\rho_2,\sigma_2') \} \quad \textbf{ELSE } F(\rho_2,\sigma_2')))$$
$$where \ \ \sigma_2' = \sigma_2[append(\sigma_2.\rho_2.\texttt{c},\ \sigma_2.\rho_2.\texttt{d})/\rho_2.\texttt{c},\ false/\rho_2.\texttt{c.connect},\ \epsilon/\rho_2.\texttt{d}]$$

$$Isconnect(\rho,\sigma) = (\exists \texttt{c} \in \textbf{ICI} : \sigma.\rho.\texttt{c.connect} = true)$$

並列実行文

$$\mathcal{C}[\![\ \mathrm{S}_1\ //\ \mathrm{S}_2\ ]\!]_g = (\ \lambda(\rho,\sigma) : \{(\rho,\ \sigma[\ \sigma_1.\rho.Var(\mathrm{S}_1)/\rho.Var(\mathrm{S}_1),\ \sigma_2.\rho.Var(\mathrm{S}_2)/\rho.Var(\mathrm{S}_2),$$
$$(w_1 \wedge \sigma_1.wait) \vee (w_2 \wedge \sigma_2.wait)/wait,\ (\sigma_1.l_0 + \sigma_2.l_0)/l_0])$$
$$|\ (\rho_i,\sigma_i) \in \mathcal{C}c[\![\mathrm{S}_i]\!]_g(\rho, \sigma[l_i/l_0]),\ i = 1,2$$
$$partition_2(\sigma, l_0) = (l_1, l_2) \qquad \})$$

ただし $w_i = \neg SKIP_\rho(\sigma, LABEL(\mathrm{S}_i)), \quad (i = 1,2)$

非決定的選択文

$$\mathcal{C}[\![\ \mathrm{S}_1\ []\ \mathrm{S}_2\ ]\!]_g = (\lambda(\rho,\sigma) : \textbf{IF } \ \sigma.wait = true$$
$$\textbf{THEN } (\textbf{IF } \neg SKIP_\rho(\sigma, LABEL(\mathrm{S}_1)) \ \textbf{THEN } \ \mathcal{C}[\![\mathrm{S}_1]\!]_g(\rho,\sigma)$$
$$\textbf{ELSE } \ \mathcal{C}[\![\mathrm{S}_2]\!]_g(\rho,\sigma))$$
$$\textbf{ELSE } \ \mathcal{C}[\![\mathrm{S}_1]\!]_g(\rho,\sigma) \cup \mathcal{C}[\![\mathrm{S}_2]\!]_g(\rho,\sigma) \qquad)$$

繰り返し文

$$\mathcal{C}[\![\texttt{while } E \texttt{ do } \mathrm{S}]\!]_g = fix(\lambda F : (\lambda(\rho,\sigma) : \textbf{IF } \ (\sigma.wait = false \wedge \mathcal{E}[\![E]\!]_\rho.\sigma = false)$$
$$\textbf{THEN } \ \{ (\rho,\sigma) \}$$
$$\textbf{ELSE } \ (\mathcal{C}[\![\mathrm{S}]\!]_g \circ$$
$$(\lambda(\rho_1,\sigma_1) : \textbf{IF } \ \sigma_1.wait = true$$
$$\textbf{THEN } \ \{(\rho_1,\sigma_1)\}$$
$$\textbf{ELSE } \ F(\rho_1,\sigma_1)))$$
$$(\rho,\sigma)) \qquad)$$

図３．意味関数（その３）

プロセス呼び出し文

$\mathcal{C}[\![L_i:\ \mathtt{p(a)}]\!]_g$

$= (\lambda(\rho, \sigma):$

IF $\sigma.wait = false$

THEN

$\{(\rho, \sigma_1'') \mid \exists(\rho_1', \sigma_1') \in g(\sigma.\rho.\mathtt{p})).(\rho,\ \sigma).(\rho.\mathtt{a}),$

$\sigma_1'' = (\mathbf{IF}\ \ \sigma_1'.wait = false$

$\mathbf{THEN}\ \ \sigma[\sigma_1'.\rho.\mathtt{a}/\rho.\mathtt{a}]$

$\mathbf{ELSE}\ \ \sigma_1'[true/\rho.\mathtt{p.wait},\ \ L_i/\rho.\mathtt{p.waitloc},\ \rho_1'/\rho.\mathtt{p.env}]\ )\ \}$

ELSE

$\{(\rho, \sigma_2'') \mid \exists(\rho_2', \sigma_2') \in (g(\sigma.\rho.\mathtt{p}))(\sigma.\rho.\mathtt{p.env},\ \sigma).(\rho.\mathtt{a}),$

$\sigma_2'' = (\mathbf{IF}\ \ \sigma_2'.wait = false$

$\mathbf{THEN}\ \sigma[\sigma_2'.\rho.\mathtt{a}/\rho.\mathtt{a},\ false/wait,\ false/\rho.\mathtt{p.wait}]$

$\mathbf{ELSE}\ \sigma_2'[true/\rho.\mathtt{p.wait},\ L_i/\rho.\mathtt{p.waitloc},\ \rho'/\rho.\mathtt{p.env}])\ \})$

図３．意味関数（その４）